Moteur de Recherche Corese
Rapport d'Activité

20/06/2006 – 20/09/2007

Virginie BOTTOLLIER – Virginie.Bottollier@sophia.inria.fr
Equipe ACACIA / EDELWEISS, INRIA Sophia-Antipolis
Contact: Olivier Corby – Olivier.Corby@sophia.inria.fr (04 92 38 78 71)

Résumé

Ce document présente les travaux réalisés depuis le dernier rapport d'activité (20/06/2006) [1] en tant qu'ingénieur associée dans l'équipe ACACIA / EDELWEISS.
Le travail a principalement consisté à intégrer le langage de requête standard SPARQL au moteur de recherche Corese et à développer celui-ci.
La première année s'est déroulée en trois parties : la réalisation d'un *Parser* (JavaCC), l'intégration de la syntaxe de Corese et l'ajout de nouvelles fonctionnalités SPARQL dans Corese. En parallèle, j'ai augmenté la base de test et participé à la gestion du projet de développement de Corese (GForge, Subversion, Java 1.5).
La deuxième année s'est articulée autour de trois grands axes: une partie « documentation » (réalisation et publication d'un manuel utilisateur, *benchmark* Corese/Jena), une partie « distribution » (création d'une nouvelle API pour les utilisateurs, distribution de la version *standalone* de Corese), et une partie plus « développement » (ajout de nouvelles fonctionnalités, développement d'un Service Web).

Table des Matières

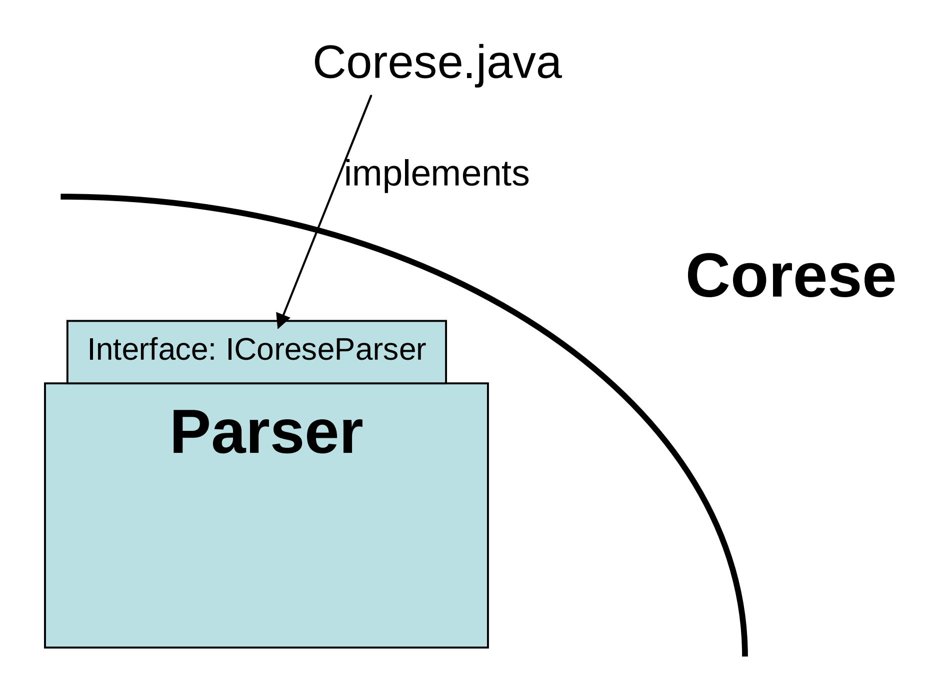
Première année (16/10/2006 – 16/10/2007)

Comme décrit dans le précédent rapport d'activité, les 8 premiers mois ont servi à intégrer SPARQL[2] dans Corese[3] (développement de la grammaire SPARQL, ajout des fonctionnalités de Corese dans cette grammaire) et à quelques taches annexes (travail sur les Datatypes, écriture de tests, passage à subversion et GForge).

Les différentes tâches prévues pour la fin de cette première année ont toutes été réalisées :

- Finaliser le parser (0.5 Mois)

Apres avoir introduit de nombreuses fonctions (fonctions d'agrégations, fonctions basées sur *XPath*…), j'ai rendu le *package* correspondant au nouveau parser (SPARQL + fonctionnalités spécifiques à Corese) indépendant du reste du code en créant plusieurs interfaces. Pour tester ce nouveau parser en dehors du reste de Corese, il suffisait d'implémenter une interface (*ICoreseParser*), puis de tester le *parsing* avec la classe contenant la méthode « main ».



- *Profiling* de Corese (1 Mois)

J'ai cherché des outils pour réaliser le *profiling* de Corese ; j'ai tout d'abord trouvé *TPTP* [4], un *plugin* pour *Eclipse* ; mais il est très gourmand en mémoire. Apres avoir réussi à réduire un peu sa lenteur en le configurant, nous avons préféré utiliser « *OptimizeIt* » [5] (soumis à licence, installé sur une des machines de l'équipe).
Avec « *OptimizeIt* », j'ai relevé le nombre d'instances, l'endroit où elles sont créées, le temps passé dans les méthodes, la couverture du code (classes non chargées, faible pourcentage de code utilisé dans les classes, classes utilisées à 100%...). Cela nous a permis de supprimer un bon nombre de classes inutiles.
Le *plugin TPTP* nous a quand même permis de trouver plusieurs méthodes inutilisées et de les effacer.

- Langage de règles : utilisation du nouveau parser (0.5 Mois)

Avec Corese, il est possible d'écrire des « règles d'inférences ». J'ai modifié le code pour que le langage de règles soit désormais proche de SPARQL (il correspond à un sous-ensemble de SPARQL, plus les fonctionnalités de Corese). J'ai également rajouté la possibilité de définir un préfixe dans les règles.

```
<cos:rule>
    <cos:if>
       PREFIX humans: <http://www.inria.fr/2007/04/17/humans.rdfs#>
       {
        ?x rdf:type humans:Female .
        ?x rdf:type humans:Person
       }
    </cos:if>
    <cos:then>
       { ?x rdf:type humans:Woman }
    </cos:then>
</cos:rule>
```

Exemple de règle : Si on trouve quelqu'un (?x) qui est de type humans:Female et humans:Person, alors cet individu (?x) est aussi de type humans:Woman

- Java 1.5 (1 Mois)

Corese est désormais codé avec Java 1.5 [6]. A part quelques noms de variables à changer (car ceux-ci sont devenus des mots clés, ex : *enum*), il n'y a pas eu de grands changements à faire pour que Corese fonctionne avec la nouvelle version. Pour utiliser au mieux les optimisations apportées, j'ai revu une bonne partie du code de Corese, en spécifiant le type des Map, Array, Vector… Cela a permis de supprimer les conversions de type et de détecter certaines erreurs de typages à la compilation.
Ex: `Vector v = new Vector();` => `Vector<String> v = new Vector<String>();`

Certaines tâches prévues pour la 2ème année ont également pu être faites :

- Intégration du nouveau parser dans Sewese[7] (0.25 Mois)

J'ai travaillé avec l'ingénieur qui s'occupe de Sewese – *framework* permettant la création simplifiée d'applications pour le web sémantique – afin d'intégrer la nouvelle version de Corese, contenant le parser SPARQL ; il a fallu réécrire plusieurs requêtes. A chaque nouvelle version de Corese, cela nous servait également de test supplémentaire et a permis la correction de plusieurs *bugs*.

- Traitement des exceptions, *refactoring* (0.75 Mois Mois)

J'ai relevé toutes les exceptions de Corese, puis je les ai regroupées, spécialisées ; certaines étaient de simples « catch (Exception e) ») d'autres ne conduisaient à aucun traitement… J'ai créé un *package* exception, et une hiérarchie des exceptions, en créant quelques nouvelles classes d'exceptions.
Avec Corese, il y a 3 façons de gérer les erreurs :

- avec des exceptions
- avec un *logger*, en imprimant un message d'erreur
- avec un tableau d'erreurs et la fonction addError(String message)

J'ai essayé de rendre cohérente et homogène la gestion des erreurs, ce qui a conduit a un *refactoring* d'une bonne partie du code.

Deuxième année (17/10/2006 – 17/10/2007)

Bilan des tâches effectuées (17/10/2006 – 20/09/2007)

- Documentation : Manuel utilisateur (+ *refactoring* et correction de *bugs*) (2 Mois)

J'ai réalisé un manuel utilisateur pour Corese [8], en anglais, actuellement en ligne et que nous mettons à jour à chaque nouvelle version de distribution.
Rédiger cette documentation nous a permis de trouver des *bugs*, que nous avons corrigés au fur et à mesure.
Plusieurs personnes de l'équipe m'ont aidée en la relisant et en m'indiquant des fautes ou des manquements.
Dans une première partie, cette documentation présente Corese, son interaction avec les différents projets européens de l'équipe, comment utiliser la version *Standalone*. Ce manuel présente également toutes les fonctionnalités non présentes dans SPARQL, avec pour chacune un ou plusieurs exemples. Puis viennent la création et l'utilisation de fonctions externes et les règles. Enfin la troisième partie, plus technique, aborde l'API de Corese, le traitement des résultats et les fichiers de configuration.

- Modularisation du code (*package projection*) (0.5 Mois)

Comme cela avait été fait pour le parser, j'ai essayé d'isoler le *package projection*. Pour cela j'ai dû créer plusieurs interfaces et deux *factories* utilisant les fonctions *« Class.forName(…) » et « newInstance(…) »* de Java.

```
try{
  Class valueClass = Class.forName("package.NomDeLaClasse");
  Object[] argClass = { arg.getClass() };
  Object[] arg = { arg };
  Object o = valueClass.getConstructor(argClass).newInstance(arg);
}
catch (Exception e) {
  e.printStackTrace();
}
```

Création d'une instance d'une classe à partir de son nom

- Création d'un service web (0.5 Mois)

Je me suis documentée sur les Web Services avec Internet. Après quelques réunions, il a été convenu d'utiliser dans un premier temps la plateforme Axis [9]. J'ai commencé par développer un service web de base pour Corese qui a été utilisé par un des projets européens de l'équipe.
Le service développé prend en paramètres une URL représentant un lien vers un fichier décrivant une ontologie, une URL vers un fichier d'annotations, une URL vers un fichier de règles, et une requête. Il est possible de donner un chemin vers un répertoire entier au lieu d'un seul fichier. Le résultat de la requête SPARQL est retourné sous la forme d'une chaîne de caractères (*XML Result Format* ou RDF).

- Utilisation du service web de Corese dans Sewese [7] (0.5 Mois)

L'ingénieur s'occupant de Sewese m'a expliqué rapidement les différentes parties de ce *framework*. J'ai rajouté une nouvelle page JSP faisant appel au service web de Corese. Cette page n'est pas actuellement utilisée, mais elle sert de « preuve de concept ».
Cela m'a permis de me familiariser avec le développement d'une application web, d'utiliser JSP, Java, les tags (tld), le fichier de configuration de d'une application web (web.xml)...

- Création d'une nouvelle API pour les utilisateurs de Corese (2 Mois)

Nous avons rendu Corese plus compréhensible aux utilisateurs en extrayant les fonctions qui leurs sont utiles dans des interfaces. Pour cela, avec l'aide de plusieurs membres de l'équipe, j'ai conçu une nouvelle couche logicielle au dessus de Corese, composée essentiellement d'interfaces et de quelques *factories*, permettant la communication avec le monde extérieur.

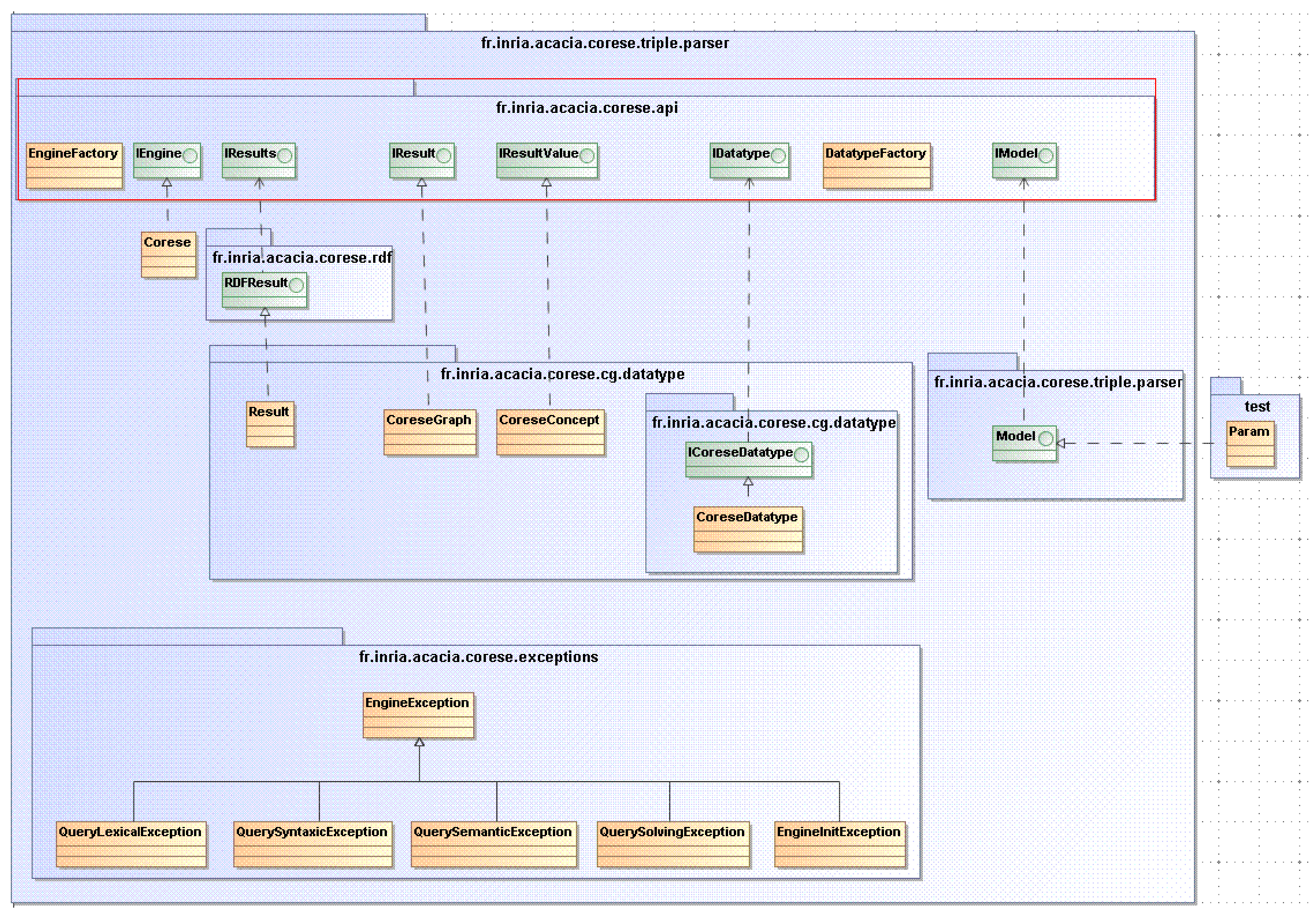


Diagramme de classe simplifié permettant de voir la nouvelle API

- Relecture des spécifications SPARQL, mise à jour de la grammaire (0.25 Mois)

SPARQL a atteint le stade de « Candidate Recommandation » depuis le 14 Juin 2007. La relecture de toute la spécification a conduit à l'ajout de nombreux tests et à la mise à jour de la grammaire (nouveau mot clé : REDUCED).

- Traceur/Débuggueur de la projection : étudier le mode *debug* d'Eclipse (0.25 Mois)

Nous n'avons malheureusement pas eu le temps de faire un vrai traceur/débuggueur comme c'était prévu initialement. Nous avons tout de même eu le temps d'étudier ce que nous offre le débuggueur d'Eclipse et les fonctionnalités poussées qu'il nous offre.
Il semblerait que ce débuggueur suffise aux besoins des développeurs.

Quelques tâches non prévues ont également été réalisées :

- Nouvelles versions de distributions : V2.2.2 (Décembre 2006) et v2.3.0 (Mai 2007) (2 * 0.5 Mois)

Pour distribuer Corese au public, nous faisons une version *standalone*. A chaque fois qu'une nouvelle version est mise en ligne, il faut tester cette nouvelle version, corriger le tutorial de Corese en rajoutant des questions/réponses sur les nouvelles fonctionnalités, corriger la documentation en ligne, réécrire la *javadoc*, écrire un fichier d'exemples permettant d'utiliser Corese…
La version de Décembre correspondait à tout le travail réalisé depuis mon arrivée ici : le nouveau parser SPARQL (d'où réécriture de certaines requêtes), le passage à Java 1.5,…
La version de Mai correspond à l'ajout d'une nouvelle API pour l'utilisateur ainsi que de quelques fonctionnalités OWL.

- Ajout de nouvelles fonctionnalités (select functions…) (1 Mois)

Après avoir étudié le code en profondeur, j'ai ajouté une nouvelle fonctionnalité à Corese : l'utilisateur peut désormais sélectionner une fonction dans la requête SPARQL. Après avoir modifié la grammaire, il a fallu ensuite faire tout le mécanisme.

Exemple de requête utilisant cette fonctionnalité :

```
SELECT ?type function(?x) as ?fun
WHERE { ?x rdf:type ?type }
```

Par la suite, j'ai réécrit une bonne partie du code en l'optimisant.

- *Benchmark* Jena/Corese [10] (0.5 Mois)

Nous avons rédigé un article portant sur les optimisations dans Corese. Afin de les mettre en évidence, j'ai réalisé une comparaison entre Jena [11] (un *framework* pour créer des applications pour le web sémantique, contenant un moteur d'inférence et utilisant SPARQL) et Corese. Le *benchmark* comporte une centaine de tests (dans environ 20% des cas, Corese est meilleur que Jena, et dans 80%, ils sont équivalents) ; il a abouti à la rédaction d'un compte-rendu publié sur le site de Corese.

- Formation sur les Services Web + Application (0.5 Mois)

J'ai suivi la formation « Développer des Services Web pour Java », pendant 4 jours à Paris début Juin. Cette formation, riche en travaux pratiques, m'a appris beaucoup de choses :

- sur les services web : les « piles » (Axis, XFire, WebServiceStudio); les langages/technologies utilisés (WSDL, SOAP, UDDI…)
- sur la culture informatique en général : découverte de .Net, rappel des principaux protocoles du Web, rappel d'UML…

J'ai également suivi une formation de mise à niveau pour Java 6.0 qui a été dispensée à l'équipe EDELWEISS les 28 et 29 Aout.

- Nettoyage de code, ajout d'un « parser de triple » (0.5 Mois)

L'équipe souhaite rendre disponible Corese en open-source. Pour cela, une partie « nettoyage du code » a été entamée, supprimant du code obsolète.
Charger de très grandes ontologies prend du temps avec Corese. Après analyse, on peut voir qu'environ 50% du temps de chargement est utilisé par ARP [12], le parser RDF/S développé par HP que nous utilisons. Pour réduire ce temps de chargement, nous avons stocké les triplets une fois générés dans un fichier (sous la forme de NTriples [13]), puis, aux chargements suivants, de parser ce fichier de triplets plutôt que celui contenant les données en RDF/S XML. Pour l'ontologie *Wordnet*[14], une grosse ontologie de plus de 400 000 triplets (il y en a 20 000 dans Corese), nous sommes passé d'un temps de chargement de 33.43s avec RDF/S à 18.54s avec le parser de triples (c'est un très bon résultat sachant que la partie « Corese » prend environ 16.50s avec cette ontologie).
Nous avons également essayé de stocker un vecteur de triplets sérialisé, mais la lecture de ce vecteur prenant trop de temps, l'idée a été abandonnée et nous sommes retournés au chargement des NTriples.

- Intégration de RDFa (0.5 Mois)

RDFa [15] est une syntaxe permettant d'intégrer des triplets RDF/S dans du XHTML. C'est ce qui va permettre l'expansion du web sémantique. Nous avons décidé d'intégrer un parser RDFa (développé par un membre de l'équipe) dans Corese, c'est-à-dire de construire et de charger des triplets à partir d'un document RDFa.

- Mise en place d’un moyen pour allouer une source à un document ou/et un triplet (0.5 Mois)

Lorsque l’on interroge un serveur RDF/S, la source des données peut être importante. Le langage SPARQL propose un moyen d’interroger cette source, or cette notion de source n’existe pas actuellement en RDF, nous avons voulu l’ajouter. Pour cela, nous proposons de rajouter un attribut dans les fichiers RDF/S permettant de préciser la source d’un triplet ou d’un ensemble de triplets.
Nous avons récupéré le code-source du parser RDF que nous utilisons (ARP) et rajouté l’attribut « cos:graph » qui permet de spécifier la source. Cette fonctionnalité a été implémentée dans Corese.
Nous envisageons de soumettre ce travail au W3C.

Exemple :

```
<rdf:RDF xmlns:dc="http://purl.org/dc/elements/1.1/"
  xmlns:foaf="http://xmlns.com/foaf/0.1/"
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:cos="http://www.inria.fr/acacia/corese#"
  cos:graph="http://www.w3.org">
 <rdf:Description rdf:about="http://www.w3.org/TR/2004/REC-rdf-mt-20040210/">
  <dc:title>RDF Semantics</dc:title>
  <dc:creator>
    <foaf:Person rdf:about="http://www.ihmc.us/users/user.php?UserID=42"
      cos:graph="http://www.ihmc.us" >
     <foaf:name>Patrick Hayes</foaf:name>
     <foaf:mbox rdf:resource="mailto:phayes@ihmc.us"/>
   </foaf:Person>
  </dc:creator>
 </rdf:Description>
</rdf:RDF>
```

Fichier RDF

```
<http://www.w3.org/TR/2004/REC-rdf-mt-20040210/> dc:title "RDF Semantics"
Source: http://www.w3.org

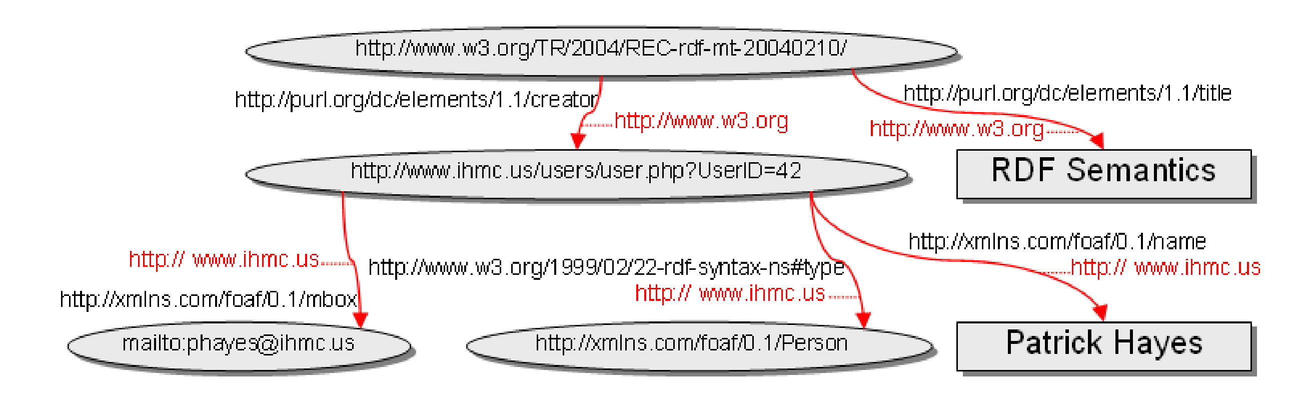
<http://www.w3.org/TR/2004/REC-rdf-mt-20040210/> dc:creator
<http://www.ihmc.us/users/user.php?UserID=42>
Source: http://www.w3.org

<http://www.ihmc.us/users/user.php?UserID=42> rdf:type foaf:Person
Source: http://www.ihmc.us

<http://www.ihmc.us/users/user.php?UserID=42> foaf:name "Patrick Hayes"
Source: http://www.ihmc.us

<http://www.ihmc.us/users/user.php?UserID=42> foaf:mbox
<mailto:phayes@ihmc.us>
Source: http://www.ihmc.us
```

Triplets générés



Représentation graphique du fichier RDF

- *Profiling* : suite (0.5 Mois)

Comme Corese est maintenant écrit en Java 1.5 et que l'équipe utilise *Eclipse* et non plus *JBuilder*, il a fallu trouver un autre outil de *profiling*, car nous n'avons pas la licence pour *OptimizeIt* avec cette configuration.
Nous avons choisi de réessayer TPTP, le *plugin Eclipse*. Ce plugin a beaucoup évolué en un an, mais il reste très gourmand en mémoire ; c'est pourquoi nous n'avons pas pu utiliser toutes les fonctionnalités souhaitées. Néanmoins, nous avons pu vérifier qu'aucun code non optimisé n'a été introduit depuis l'étude de l'année précédente.

Planning pour la fin de l'année (20/09/2007 – 17/10/2007)

- Mise en place du *SPARQL Protocol* [16], finalisation des services web de Corese (0.5 Mois)

Nous souhaitons mettre en place le protocole SPARQL permettant une présentation standardisée du web service de Corese. Cela me permettra également d'appliquer ce que j'ai appris lors de ma formation,

- Analyse/conception *plugin Eclipse* SPARQL (0.5 Mois)

Enfin, si le temps le permet, nous aimerions effectuer l'analyse d'un *plugin Eclipse* pour le langage SPARQL.

Perspectives

Si le travail d'analyse d'un plugin Eclipse pour le langage SPARQL se révèle intéressant, celui-ci devrait être mis en place.

L'équipe EDELWEISS envisage de construire une nouvelle plateforme de graphes, en partenariat avec une équipe de Montpellier. Pour cela, la « Color Griwes » a été créée [17]. Corese va quand même continuer à être maintenu car il est utilisé dans plusieurs projets européens.

Le projet RIF du W3C (*Rule Interchange Format*) [18] avançant très lentement, nous n'avons pas encore pu l'intégrer à Corese.

Bénéfice Personnel

Au cours de ces deux années, j'ai pu acquérir une formation et une expérience conséquente et me perfectionner dans les domaines suivants :

1. Organisation
   - Création d'un planning, avec un bilan mensuel des objectifs réalisés / à réaliser
   - Travail en équipe
   - Travail dans un contexte de recherche
   - Connaissance du W3C

2. Outils
   - Eclipse, Ant
   - JUnit, tests de non régressions
   - Subversion, tortoiseSVN
   - MagicDraw, OptimizeIt, Unix, TPTP

3. Technologies
   - Le Web Sémantique : RDF/S, OWL, SPARQL, les ontologies
   - Langages : Java, javaCC
   - Création d'un parser à partir d'une grammaire
   - Les Web Services, Axis, XFire, WSDL
   - Conception et développement d'une API
   - Sensibilisation aux problèmes de « compatibilité ascendante »

Conclusion

Pour résumer, j'ai permis à Corese d'utiliser le futur standard du W3C SPARQL, j'ai participé au développement de Corese en l'installant sur GForge avec subversion ; j'ai également beaucoup augmenté la base de tests. Deux apports majeurs pour les utilisateurs consistent dans la réalisation d'une documentation en ligne et la création d'une nouvelle API. Enfin, j'ai aussi permis l'ajout de nouvelles fonctionnalités et le développement d'un service web.
D'un point de vue plus personnel, ces deux années au sein de l'INRIA m'ont permis d'acquérir de l'expérience en développement informatique, de renforcer mes compétences en Java en me familiarisant avec Eclipse. J'ai également découvert les technologies du web sémantique. Travailler dans un centre de recherche tel que l'INRIA m'a donné une ouverture d'esprit sur le monde informatique (cours, séminaires, colloquiums…).
J'ai beaucoup apprécié de travailler au sein de l'équipe EDELWEISS que je remercie chaleureusement pour son accueil et sa confiance pendant ces deux années. Enfin, je tiens à remercier plus particulièrement Olivier Corby et David Rey, mes encadrants scientifique et technique.

Références

[1] Rapport d'activité 2006
http://www-sop.inria.fr/dream/rapports/suivi-dev/2005/ra-acacia.pdf

[2] SPARQL
http://www.w3.org/TR/rdf-sparql-query/

[3] Corese
http://www-sop.inria.fr/acacia/corese/

[4] TPTP
http://www.eclipse.org/tptp/

[5] OptimizeIt
http://www.borland.com/downloads/download_optimizeit.html

[6] Java 1.5
http://java.sun.com/j2se/1.5.0/

[7] Sewese
http://www-sop.inria.fr/acacia/soft/sewese/

[8] Manuel d'utilisation de Corese
http://www-sop.inria.fr/acacia/soft/corese/manual/

[9] Axis
http://ws.apache.org/axis/

[10] Benchmark Corese/Jena
http://www-sop.inria.fr/acacia/corese/test.html

[11] Jena
http://jena.sourceforge.net/

[12] ARP
http://www.hpl.hp.com/personal/jjc/arp/

[13] NTriples
http://www.w3.org/TR/rdf-testcases/#ntriples

[14] Wordnet
http://www.semanticweb.org/library/

[15] RDFa
http://www.w3.org/TR/xhtml-rdfa-primer/

[16] SPARQL Protocol
http://www.w3.org/TR/rdf-sparql-protocol/

[17] Color Griwes
http://www-sop.inria.fr/acacia/project/griwes/wakka.php?wiki=ColorGriwes

[18] RIF
http://www.w3.org/2005/rules/